CONV3TO-PC/2ØØØ Ver. 1.Ø

USER MANUAL

Educational Micro Systems Inc., P.O. Box 471, Chester, N.J. 07930 • 201-879-5982

IMPORTANT NOTICE

CONV3TOPC and CONV3TO2ØØØ are distributed on an "AS IS" basis only and without warranty. EMSI makes no express or implied warranty of any kind with regard to these programs including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Neither EMSI nor any authorized EMSI dealer shall have liability or responsibility to any person or entity with respect to any liability, loss, or damage caused or alleged to be caused by CONV3TOPC or CONV3TO2ØØØ, including, but not limited to, any interruption of service, loss of business or anticipatory profits, or consequential damages resulting from the use of these programs.

WARNING

Good programming practice dictates that new programs should be thoroughly tested before being put to use. Likewise, every Mod III program that is converted with CONV3TOPC or CONV3TO2ØØØ should be completely re-tested. Never assume that a converted program will perform in exactly the same manner as it did on the Mod III just because the converted program meets the syntax requirements of a different microcomputer. Parallel testing with contrived test data and parallel production runs with live data should be conducted for a period of time sufficient to ensure that results are identical. Always save the Mod III version of converted programs and the files that they maintained in the event that bugs are discovered in the converted programs at a later date.

CONV3TOPC and CONV3TO2ØØØ automate much of the work required to produce IBM PC and Tandy 2ØØØ programs which are syntactically correct. The burden of testing converted programs rests with the user. Be sure to read Appendix D for more information on testing.

CONV3TOPC Ver. 2.Ø

CONV3TOPC Ver. 2.Ø replaces CONV3TOPC Ver. 1.Ø and CONV3TO2ØØØ Ver. 1.Ø. It runs on any IBM PC or compatible using PC/MS-DOS and BASICA or GW-BASIC, including the Tandy 1ØØØ, 12ØØHD and 2ØØØ. Please ignore all references to CONV3TO2ØØØ and CNV32ØØØ.BAS in the Ver. 1.Ø user manual.

Ver. 2.Ø Enhancements

1. Function keys:
   F1 is active: ABORTS conversion in progress.
   F2 through F14 (or F16) are inactive.

2. USING statement: M3-BASIC's "%" and "[" symbols are automatically replaced with GW-BASIC's "\" and "^" symbols.

3. FIELD statement: Each occurrence of "AS" is delimited with spaces.

4. New MENU: When a conversion run is completed or terminated by you via the F1 key, Ver. 2.Ø presents a menu from which you may choose to: Re-run CNV3TOPC.BAS, Exit to BASIC, Exit to PC/MS-DOS, or load the converted program for manual revisions.

5. Seven additional keywords are flagged: INPUT@, KILL, RUN, LOAD, LOF( ), FIELD and USING.

6. Reports: Each page is numbered and contains a descriptive heading. If you select menu options D, E, or F requesting printed output, CNV3TOPC.BAS will prompt you to supply information about how your printer is configured. Your responses depend on printer model, pitch, lines per inch and paper length settings.

1. Characters per line (default = 8Ø)?

If your printer is set for 1Ø pitch and limited to narrow 8½ inch paper you can most likely just press <ENTER>. Refer to your printer manual for the appropriate reponse. For example, if you have an OKIDATA ML92 set for 12 pitch, type 96 and press <ENTER>.

2. Lines per page (default = 66)?

If your printer is set to print 6 lines per inch on standard 11 inch paper just press <ENTER>, otherwise type the appropriate reponse. For example, if your printer is set for 8 lines per inch and 11 inch paper, type 88 and press <ENTER>.

Note that you may permanently change the displayed default values if they do not agree with the way you typically configure your printer - load CNV3TOPC.BAS and edit lines 6521Ø and 6522Ø per your requirements. Further, if you always use the displayed default values, you may eliminate the need to respond to above prompts by activating line 65225.

TABLE OF CONTENTS

Chapter 1 INTRODUCTION

1.1 REGISTRATION.

Please fill out and mail the software registration card as soon as possible since this is our only way of knowing you are a licensed CONV3TOPC or CONV3TO2ØØØ user. We will endeavor to notify licensed users about enhancements and new releases.

1.2 Trademark Credits

Radio Shack, Tandy 2ØØØ, TRS-8Ø, TRS-8Ø Mod III, and TRSDOS 1.3 are trademarks of Radio Shack, a division of Tandy Corporation. MS-DOS and GW-BASIC are trademarks of MICROSOFT Corporation. IBM PC is a trademark of International Business Machines Corporation.

1.3 User Manual Terminology

This manual is provided for both CONV3TOPC and CONV3TO2ØØØ licensed users. Because the majority of information presented herein applies to both software packages and machines, the following terms and definitions will be used for convenience of understanding:

CV3	refers to both CONV3TOPC and CONV3TO2ØØØ. For example, "Place the CV3 diskette in drive B" means the same as: "Place the CONV3TOPC diskette in drive B" and "Place the CONV3TO2ØØØ diskette in drive B".
PC	refers to both the IBM PC and Tandy 2ØØØ.
MS-DOS	refers to both PC-DOS and MS-DOS.
GW-BASIC	refers to BASIC as implemented on the PC.
M3-BASIC	refers to BASIC as implemented on the TRS-8Ø Model III.
LINK3PC	the generic name used to refer to the file transfer utility program enclosed with this package on a separate diskette.

1.4 CV3 Overview

Thousands of TRS-8Ø Mod III computers were sold before 1984, and thousands of programs have been bought by Mod III users. With the Mod III's discontinuation and the growing popularity of new 16-Bit machines like the IBM PC and its compatibles, many Mod III users have considered upgrading to the new machines but resisted doing so because Mod III software is not directly compatible. EMSI's two powerful utilities, CONV3TOPC and CONV3TO2ØØØ, solve this problem by providing a quick, easy, and accurate means to convert your Mod III software to the IBM PC compatibles.

Without the assistance provided by these packages you would have to convert your programs through a long and tedious process. For instance, GW-BASIC keywords require space delimiters while M3-BASIC does not:

```
      M3-BASIC                    GW-BASIC
1Ø PRINTA$                  1Ø PRINT A$
2Ø FORI=1TO2ØSTEP2          2Ø FOR I=1 TO 2Ø STEP 2
3Ø IFA=BTHEN4ØØ             3Ø IF A=B THEN 4ØØ
4Ø NEXTI                    4Ø NEXT I
```

The other numerous changes performed for you include: replacement of the PRINT@ instruction with LOCATE and PRINT; adjustment of TAB addresses, and replacement of exponential symbols.

The step by step procedure for converting a M3-BASIC program to GW-BASIC is explained later in this manual. In general, the procedure requires four steps:

1. Prepare the Mod III program.
    A. Save program in ASCII format.
    B. Validate ASCII version.
2. Transfer ASCII version to MS-DOS diskette on Mod III.
    A. Format MS-DOS diskette.
    B. Copy Mod III ASCII version to MS-DOS diskette.
3. Run conversion program on PC.
4. Manual revision.
    A. Edit flagged lines.
    B. Merge subroutines as required.

CONV3TOPC and CONV3TO2ØØØ were written to save you time by automating most of the conversion and flagging program lines which need manual attention. Flagged conditions and their GW-BASIC alternatives are explained in this user manual.

EMSI has tried to provide a comprehensive, error free user manual and bug free software. If you find something you believe we overlooked, we would sincerely appreciate your suggestions.

Chapter 2 PROGRAM OPERATION

2.1 Preparing CV3 Master Copies

CV3 is written in GW-BASIC and is provided on a single sided, 40 track diskette readable by both the IBM PC and the Tandy 2000. Note, however, that the diskette does not contain MS-DOS (it is not an MS-DOS system diskette). Since it is not good practice to put a purchased diskette into daily use, we suggest that you create personal "working copies" to aid your use of the program while the original is stored away in a safe place.

The following assumes you have a PC with two floppy disk drives. Those with only one floppy drive or a hard drive should review their MS-DOS manuals for slight variations in the procedure.

If you do not have a hard drive, we suggest that you place your working copies of CV3 on MS-DOS system diskettes. Use the FORMAT command to prepare a few blank diskettes, use DISKCOPY or COMPDUPE to create duplicate MS-DOS masters, and use COPY to transfer the contents of your CV3 diskette to your working copy.

Power up your PC, place an MS-DOS master diskette in drive A, and press the RESET button or multi-key equivalent. Be sure there is a write protect label on the MS-DOS master to avoid accidents. Answer the date and time prompts and then:

1. Place a blank diskette in drive B.

2. Execute the FORMAT command. Type **FORMAT B:** and press <ENTER>.

3. Execute the DISKCOPY command. Type **DISKCOPY A: B:** and press <ENTER>.

4. Remove your original MS-DOS master from drive A and store away.

5. Remove your MS-DOS copy from drive B and place it in drive A.

6. Place your CV3 diskette in drive B.

7. Execute the COPY command.
   Type: **COPY B:CNV3TOPC.BAS A:** and press <ENTER> (for IBM PC users), or
   Type: **COPY B:CNV32000.BAS A:** and press <ENTER> (for Tandy 2000 users).
   Please note that the actual program names begin with **CNV** not **CONV**.

8. Optionally, execute the COPY command: **COPY B:*.ASC A:** A few short GW-BASIC subroutines, which may be required by your converted programs, are stored on the CV3 diskette in ASCII format for easy MERGEing (see Appendix G).

When step 7 or 8 is complete, you will have your first working copy. You may want to place a write protect label on this diskette, call it your "master working copy," and only use it for making additional working copies for daily use. This should eliminate your need to handle the original CV3 diskette again.

2.2 Preparing LINK3PC Master Copies

As mentioned in the CV3 Overview, this package includes a separate diskette containing a program to transfer your Mod III programs and files to a PC. Refer to Appendix H for details about the use of this program.

2.3 Preparing the Mod III program for conversion

Prior to running the CV3 conversion program, each Mod III program must (1) be saved in ASCII form, and (2) be transferred to an MS-DOS 1.Ø diskette.

SAVING THE PROGRAM IN ASCII: Power up a Mod III with TRSDOS 1.3 in drive Ø and press reset. Answer the date and time prompts. Insert the diskette containing the program to be converted in drive 1. Type BASIC and press <ENTER>. Answer the memory size and number of files questions by pressing <ENTER>. Use the LOAD command to load the program into memory.

> Type: LOAD"NAME:1" and press <ENTER>.
> Where NAME is program's filespec and 1 is the number of the drive in which it is located.

When the screen displays READY, the program has loaded and you are ready to save it in ASCII form by using the SAVE command with the "A" option (we strongly suggest reading Appendix A first).

> Type: SAVE"NAME:#",A and press <ENTER>.
> Where NAME is the new filespec you want to assign to the ASCII version of the program and # is the number of the drive where you want the ASCII version to be recorded.

We suggest that you specify a file name for the ASCII version different from the compressed version. For example, if the original Mod III program NAME was "MOD3PROG" you may want to name the ASCII version "MOD3PROG/ASC".

After successfully saving an ASCII version of the Mod III program on disk, you are ready to transfer it to an MS-DOS 1.Ø formatted diskette for easy transition to your PC.

TRANSFERRING THE ASCII VERSION TO AN MS-DOS DISKETTE: Please refer to Appendix H.

2.4 Running CV3 - CONV3TOPC and CONV3TO2ØØØ

Before running the conversion be sure you have made a working copy of the CV3 program, prepared a TRSDOS 1.3 ASCII version of the Mod III program, used the LINK3PC file transfer program to copy the TRSDOS 1.3 ASCII version to an MS-DOS diskette, and reviewed Appendices A and D.

Power up your PC, insert the MS-DOS System master containing your version of CV3 in drive A, and press the RESET button or multi-key equivalent. Answer the date and time prompts as requested. Type BASIC and press <ENTER>. Load and run CV3.

Type: RUN"A:CNV3TOPC.BAS" and press <ENTER>. (for IBM PC users), or
Type: RUN"A:CNV32ØØØ.BAS" and press <ENTER>. (for Tandy 2ØØØ users).
Please note: the actual program names begin with CNV not CONV.

If you have a printer and you plan to select options D, E, or F, then make sure the printer is on and the paper is aligned so that the perf is just above the print head.

If you plan to select option I (SAVE converted PC version on disk), then you must decide which drive will receive the GW-BASIC program. Place a formatted diskette in that drive.

Menu Selections

There are 12 menu selections from which you may choose any combination. To select an option simply type the corresponding letter. Type the letters for all selections, in any order, and then press <ENTER>. It is not necessary to separate menu selections with commas.

A. **Display errors on SCREEN.** Primarily for users without line printers, this option displays all error conditions on the screen. If you do not select option D we suggest that you select option G to pause execution so you can record errors before they scroll off the screen.

B. **Display Mod III program lines on SCREEN.** Primarily for users without line printers, this option will display each Mod III program line to be converted. Used in conjunction with A and C, it facilitates manual corrections of unresolved conditions.

C. **Display PC program lines on SCREEN.** Primarily for those without line printers, this option will display each converted GW-BASIC program line. Used in conjunction with A and B, it facilitates manual corrections of unresolved conditions.

D. **List errors on PRINTER.** All error conditions are listed on the printer if this option is selected. It is suggested for those with printers.

E. **List Mod III program lines on PRINTER.** Each Mod III program line is listed on the printer with this option. It is especially useful if you do not already have a printed listing of the Mod III program. When used in conjunction with options D and F, it assists with manual corrections of unresolved conditions.

F. **List PC program lines on PRINTER.** This option lists each converted GW-BASIC program line on the printer. If you have a printer this option is suggested. When used in conjunction with options D and E, it facilitates manual corrections of unresolved conditions.

G. **PAUSE after each error condition.** Primarily for users without line printers, this option pauses program execution after the discovery of each error to allow for manual recording. G is not suggested for those with line printers because of the extra time involved in its use.

H. **SOUND BELL after each error condition.** Mainly for those without line printers, this option beeps the PC bell for a few seconds each time an error condition is encountered. H is not suggested for those with line printers.

I. **SAVE converted PC version on disk.** The converted program is stored on disk with this option. It must be selected to save a disk version of the converted program, although flagged error conditions still must be resolved before you attempt to RUN the program. You will be prompted to enter the filespec (ie. drive and name) for the converted GW-BASIC program. The filespec must be different from that which you assigned to the Mod III program.

J. **Insert line feed after each ":"** This option inserts a down feed character and indent spaces after each colon in lines containing more than one instruction (refer to note concerning use at top of next page).

With option J
```
200  CLS:FORI=1TO3:PRINT"Option J":NEXT
```

would become
```
200  CLS:
     FOR I=1 TO 3:
     PRINT"Option J":
     NEXT
```

CV3 attempts to make all other GW-BASIC conversions before executing this option. Down feeds are inserted before spaces, and if the maximum number of characters per line, 249, is met or exceeded before execution is complete, some of the instructions will not be staggered or indented (see Appendix A).

K. **Insert line feed before IF, THEN, ELSE.** This option inserts a down feed character and indent spaces before each IF, THEN, or ELSE statement that is found (refer to note concerning use at top of next page).

With option K
```
300  CLS:IFA=BTHENPRINT"Option K"ELSEGOTO500
```

would become
```
300  CLS:
     IF A=B
     THEN PRINT"Option K"
     ELSE GOTO 500
```

CV3 inserts down feeds while it is making other GW-BASIC conversions and inserts indent spaces after all other conversions have been made. If the maximum number of characters per line, 249, is met or exceeded before execution is complete, some of the IF, THEN, or ELSE statements may not be staggered or indented (see Appendix A).

Notes concerning use of options J and K: Used together, options J, K, and F produce a GW-BASIC program listing that is very easy to read. However, we do not recommend that option J or K be selected with option I. Options J and K slow the conversion, increase the size of the resultant GW-BASIC program, and cause GW-BASIC to insert a hard line feed (ASCII 13) after each downfeed, which may cause problems when executing the converted program. If you desire a more legible listing it is better to run the conversion once with J and K (without I) and once with I (without J or K).

L. **Remove REM's and comments from PC version.** CV3 classifies two types of remark statements. One type is the **REM statement** where the word REM or the shorthand "'" is the first and only statement on the line. For example:

```
1ØØ REM*** This is a REM statement ***
```

The second type is the **REM comment**, a REM or "'" which is not the first statement on the line. For example:

```
2ØØ AV=OH+OO-AL  :  REM*** This is a REM comment ***
```

Only the text is removed from REM statements. The line number and keyword remain in case the line is referenced with a GOTO or GOSUB. The keyword and text are removed from REM comments. For example:

```
line 1ØØ becomes              1ØØ '
and line 2ØØ becomes          2ØØ AV=OH+OO-AL
```

You will be asked for the line number at which "removal" should begin.

After choosing program options and pressing <ENTER>, you will be given a review of your selections and an opportunity to revise them.

When you are satisfied with the options selected, you will be prompted to enter the filespec (drive designation and name) for the Mod III program to be converted. Type in the proper filespec and press <ENTER>. If your response is syntactically incorrect or the requested program is not found on the drive that you specified, you will be asked to enter the filespec again.

If you selected option I, you will be prompted to enter the filespec for the disk version of the converted GW-BASIC program (see option I). If the filespec you choose is that of an existing file, you will be asked for verification.

Finally, you will be reminded to ready your printer, and the conversion will begin.

Chapter 3 CV3 ERROR MESSAGES

ERROR 1 and 2 reserved.

ERROR 3 - PEEK/POKE statement encountered. Left unchanged: Refer to Appendix B for details and assistance.

ERROR 4 - Not supported: The Mod III screen supported low res graphics. Programs could access any one of 6144 screen pixels arranged in a rectangular grid consisting of 128 columns numbered Ø-127 from left to right, and 48 rows numbered Ø-47 from top to bottom. Pixels could be turned on (lit up) with the SET(C,R) command, turned off (darkened) with the RESET(C,R) command, and tested to see if they were on or off with the POINT(C,R) command.

POINT is nearly impossible to replace, but the PRINT@ instruction is an alternative for SET and RESET under certain conditions. For example, SET was often used to draw borders around the Mod III screen. You may find something like this:

```
2ØØ FOR X=Ø TO 127:SET(X,Ø):SET(X,47):NEXT
21Ø FOR Y=Ø TO  47:SET(Ø,Y):SET(1,Y):SET(126,Y):SET(127,Y):NEXT
```

For an attractive GW-BASIC substitute, see Appendix G, R6517Ø.ASC.

ERROR 5 - Mod 3 PRINT@ syntax error. Comma missing after address: This error is very unlikely because it would have caused a M3-BASIC syntax error. Before replacing PRINT@ addresses with LOCATE ROW,COL, etc., CV3 looks for the address terminator, a comma. If no comma is found, the address is not modified. Read the description for ERROR 6 to determine how to manually edit this instruction.

ERROR 6 - Mod 3 PRINT@ address contains variable(s): Most likely, this is a warning that you will be required to MERGE R65Ø3Ø.ASC with your GW-BASIC program after the conversion. Your program may contain two types of PRINT@ statements: "hard" PRINT@s with a constant address (PRINT@45Ø,"Something"), and "soft" PRINT@s with a variable address determined at execution time (PRINT@XY,"Something"). Conversion of both types is automatic unless the process would create a GW-BASIC line that is too long.

```
                PRINT@45Ø,"Something"
would be changed to:    LOCATE 8,3:PRINT"Something"
```

If the conversion of a "hard" PRINT@ is bypassed, split the line and make the conversion manually. For PRINT@A, LOCATE coordinates R(ow) and C(olumn) are determined as follows: R=INT(A/64)+1 and C=A-64*INT(A/64)+1.

Automatic conversion of "soft" PRINT@s usually does not cause a problem:

```
                PRINT@XY,"Something"
would be changed to:    ZOG%=XY:GOSUB 65Ø3Ø:PRINT"Something"
```

If the conversion of a "soft" PRINT@ is bypassed, split the line and convert as above. Remove PRINT@ and replace with:

ZOG%=(Address):GOSUB 65Ø3Ø:PRINT where (Address) is the PRINT@(Address).

If the variable address contains a comma, the line will be converted incorrectly and require manual editing. The M3-BASIC line:

```
PRINT@X(Y,3),"X(Y,3) - contains a comma"
```

will convert erroneously as follows:

```
ZOG%=X(Y:GOSUB 65Ø3Ø:PRINT 3),"X(Y,3) - contains a comma"
```

Notice that the embedded comma caused CV3 to spl-it the address. Manually edit to produce the correct result:

```
ZOG%=X(Y,3):GOSUB 65Ø3Ø:PRINT"X(Y,3) - contains a comma"
```

Note that if your program contains one or more "soft" PRINT@ instructions, you will need to MERGE (or enter from keyboard) the program R65Ø3Ø.ASC (see Appendix G).

ERROR 7 - Missing IF or THEN statement: M3-BASIC sometimes permitted THEN to be omitted from IF/THEN statements, but GW-BASIC requires a THEN with each IF. Edit the line and insert THEN in the proper place. This error will also be displayed in the unlikely situation of a missing IF.

ERROR 8 - CMD" " statement encountered: M3-BASIC supported a series of CMD"?" statements. Although GW-BASIC does not support CMD" " commands, a few may be replaced by alternate methods (see Appendix C).

ERROR 9 - Line TOO LONG: A program line being converted exceeded or would have exceeded maximum line length. Some portion of the line may not have been converted and/or some portion may have been truncated. If option I was in effect the line was written to disk as usual. Review the contents of the M3-BASIC and GW-BASIC lines to see whether manual editing is required.

ERROR 1Ø - TIME$ statement encountered: In M3-BASIC, TIME$ returns both the date and the time as a 17 byte string - "MM/DD/YY_HH:MM:SS". GW-BASIC has separate instructions for this purpose: DATE$ returns the 1Ø byte string "MM-DD-19YY" and TIME$ returns the eight byte string "HH:MM:SS". The following lines show a few M3-BASIC instructions on the left and their GW-BASIC equivalent on the right:

```
        M3-BASIC                    GW-BASIC (close) equivalent
25 A$=TIME$                 25 A$=DATE$+" "+TIME$
3Ø B$=LEFT$(TIME$,8)        3Ø B$=DATE$
4Ø C$=RIGHT$(TIME$,8)       4Ø C$=TIME$
5Ø PRINT TIME$              5Ø PRINT DATE$" "TIME$
```

The exact GW-BASIC equivalent of the M3-BASIC statement LEFT$(TIME$,8) is:

```
LEFT$(DATE$,2)+"/"+MID$(DATE$,4,2)+"/"+RIGHT$(DATE$,2).
```

Conversion becomes cumbersome if LEFT$(DATE$,8) occurs several times throughout your program. You may want to define the exact GW-BASIC equivalent as a string function at the beginning of your program. For example:

```
DEF FNDTT$=LEFT$(DATE$,2)+"/"+MID$(DATE$,4,2)+"/"+RIGHT$(DATE$,2)
```

Then simply substitute FNDTT$ for each occurrance of LEFT$(DATE$,8) or MID$(DATE$,1,8).

ERROR 11 - CINT( ) statement encountered: M3-BASIC truncates decimals, while GW-BASIC rounds decimals. For example, CINT(6.5) yields 6 with M3-BASIC but 7 with GW-BASIC. Some visual analysis of the program is required to determine if this will cause a problem. See if INT( ) can be used instead.

ERROR 12 - CLEAR statement encountered: Both M3-BASIC and GW-BASIC use this instruction to null strings, zero variables, and close open files. In M3-BASIC, if CLEAR is followed by a number it is also allocating string space. Since GW-BASIC allocates string space dynamically, any number or variable following the CLEAR instruction must be removed.

ERROR 13 - OUT statement encountered: Both M3-BASIC and GW-BASIC use OUT to send a byte to a port. However, Mod III and PC port designations differ. Refer to the Mod III and PC technical manuals to determine the required revision.

ERROR 14 - INP() statement encountered: Both M3-BASIC and GW-BASIC use INP() to retrieve a byte from a port. However, Mod III and PC port designations differ. Refer to the Mod III and PC technical manuals to determine the required revision.

ERROR 15 - ERR statement encountered: If an ON ERROR GOTO is active in a BASIC program, the interpreter will branch to the specified line number rather than abort program execution. M3-BASIC programs can test the value of ERR or ERR/2+1 in order to check the error type which was encountered. Since use of ON ERROR GOTO and ERR are common error trapping techniques, you may find, for example:

```
3ØØØ IF (ERR/2+1)= ## THEN .......
3ØØØ IF ERR = ## THEN .......
```

Note that because the Mod III assigns a false error code to ERR, you must divide by 2 and add 1 to determine the true error code. Both of the following M3-BASIC statements test whether the true error code is 13:

```
3ØØØ IF (ERR/2+1) = 13 THEN .......
3ØØØ IF ERR = 24 THEN .......
```

The GW-BASIC interpreter does assign the true error code to ERR. A GW-BASIC line to test error code 13 would be:

```
3ØØØ IF ERR = 13 THEN .......
```

If the M3-BASIC line is something like:

```
3ØØØ IF (ERR/2+1) = ## THEN .......
```

just replace the "ERR/2+1" with "ERR" because the line is already testing for the true error code. If the M3-BASIC line is something like this:

```
3ØØØ IF ERR = X THEN .......
```

replace 'X' with the true error code which is X/2+1.

Finally, some true error codes returned by M3-BASIC and GW-BASIC are not the same (see Appendix E for details).

ERROR 16 - Mod 3 TAB( ) syntax error. Right paren. missing: This error is unlikely because it would have caused a M3-BASIC syntax error. Edit the line, insert the required ")" and the equivalent GW-BASIC TAB location.

To determine the equivalent GW-BASIC TAB location, first calculate the modulo 128 equivalent of the M3-BASIC TAB address shown. Given an address of X, the actual address equals X-(INT(X/128)*128). This will produce a value between Ø and 127.

For TAB's preceded by LPRINT, or for TAB's preceded by PRINT with modulo addresses between Ø and 63, just add 1 to get the GW-BASIC equivalent. For instance:

M3-BASIC	GW-BASIC equivalent
PRINT TAB(45)"...	PRINT TAB(46)"...
PRINT TAB(128)"...	PRINT TAB(1)"...
PRINT TAB(15Ø)"...	PRINT TAB(23)"...
LPRINT TAB(9Ø)"...	LPRINT TAB(91)"...

For TAB's preceded by PRINT with addresses between 64 and 127, see ERROR 19.

ERROR 17 - Mod 3 TAB ( ) address contains variables: The conversion program does not attempt to convert TAB addresses when they contain variables. A TAB statement such as TAB(X) or TAB(R+15) will be left unchanged. Review program logic, determine what value or range of values the TAB modulo 128 address would be, and refer to ERROR 16 and 19 for assistance.

ERROR 18 - Mod 3 TAB syntax error, TAB( ) not preceded by PRINT or LPRINT: This error is unlikely because it would have caused a M3-BASIC syntax error. Edit the line, insert the required PRINT or LPRINT and refer to ERROR 16, 17, and 19 for assistance.

ERROR 19 - PRINT TAB( ) encountered with address between 64 and 127 inclusive: Fortunately, most programmers do not use TAB's between 64 and 127 so you most likely will not encounter this error flag. Given that the Mod III screen is 64 characters wide and TAB addresses are modulo 128, if the cursor is in position Ø on a line, then a PRINT TAB( ) statement can move the cursor to any position on the current line (Ø-63) as well as the next line (64-127). The sample M3-BASIC lines which follow on the left are restated in equivalent M3-BASIC lines on the right.

```
5ØØØ PRINTTAB(64)"ABC"        5ØØØ PRINT:PRINTTAB(Ø)"ABC"
51ØØ PRINTTAB(78)"ABC"        51ØØ PRINT:PRINTTAB(14)"ABC"
```

In general, if the cursor is between Ø and 63, a PRINT TAB(X) where X is between 64 and 127 may be avoided by inserting a PRINT before the TAB and subtracting 64 from the existing TAB address. Hence, if you find a M3-BASIC line such as:

```
5ØØØ PRINTTAB(85)"ABC"
```

change it to the M3-BASIC equivalent:

```
5ØØØ PRINT:PRINTTAB(21)"ABC"
```

and then add 1 to produce the GW-BASIC line:

```
5ØØØ PRINT:PRINT TAB(22)"ABC".
```

Suppose CV3 ERROR 19 is displayed for lines 4ØØØ and 4Ø2Ø in the following M3-BASIC program:

```
4ØØØ CLS:PRINTTAB(85)"ABC";
4Ø2Ø PRINTTAB(94)"DEF"
```

How do you convert these lines? Change line 4ØØØ to:

```
4ØØØ CLS:PRINT:PRINT TAB(22)"ABC";
```

In changing the TAB address of 4Ø2Ø you should subtract 64 from 94 and add 1 (i.e. subtract 63). Do not insert an extra PRINT in 4Ø2Ø because that step was accounted for in line 4ØØØ.

The GW-BASIC equivalent of lines 4ØØØ and 4Ø2Ø is:

```
4ØØØ CLS:PRINT:PRINT TAB(22)"ABC";
4Ø2Ø PRINT TAB(31)"DEF"
```

If in testing your converted program you find that the display in GW-BASIC differs from M3-BASIC, it may be because portions of the displayed lines which would automatically wrap around on the Mod III's 64 column display continue to print on the same line for up to 16 additional spaces on the PC's 8Ø column display. In addition, GW-BASIC checks to see how much space is available before displaying any text on a line. If the text length exceeds the remaining space, the GW-BASIC cursor skips to the first position of the next line before displaying any of the text.

ERROR 2Ø - USR or DEF USR statement encountered: The DEF USR statement stores the starting address of a machine language subroutine which will subsequently be called by a related USR statement. The potential problem here is not with syntax, but with the address being assigned or called.

Mod III ROM routines are not available to the PC. If the assignment or call is to one of these routines, the only solution is to write a comparable PC machine language routine, store it in protected memory, and change the address assigned to the DEF USR statement.

If the assignment or call is to a RAM routine loaded or POKEd into memory by a user program, the command syntax may not need to be changed. Nevertheless, the machine language program which is poked into upper memory must be converted. Refer to Appendix B, M3-BASIC PEEKs and POKEs, page B.4.

ERROR 21 - NEXT encountered in conditional statement: M3-BASIC, unlike GW-BASIC, allowed statements such as:

```
6ØØØ IF A=B THEN NEXT
```

and

```
6ØØØ IF A=B THEN PRINT"A=B":NEXT ELSE NEXT
```

Add a separate NEXT line at the bottom of the FOR/NEXT loop (if not already there) and replace the conditional NEXT with a GOTO. For instance:

```
6ØØØ IF A=B THEN 62ØØ
  :
62ØØ NEXT
```

(or)

```
6ØØØ IF A=B THEN PRINT"A=B":GOTO 62ØØ
  :
62ØØ NEXT
```

Note that ERROR 21 may be displayed if the NEXT found in a conditional statement is actually part of a conditional FOR/NEXT pair. For example:

```
6ØØØ IF A=B THEN FOR I=1 TO 3:
          PRINT"Ignore this false ERROR 21 flag":NEXT
```

ERROR 22 - Cassette input/output statement encountered: INPUT#-? or PRINT#-? caused this flag. These M3-BASIC instructions are used to read data from a cassette and to store data on a cassette, respectively. Because cassette processing is not supported by GW-BASIC, try to replace cassette file Input/Output with sequential disk file Input/Output. Review the INPUT#, PRINT#, OPEN, CLOSE, and EOF() instructions in GW-BASIC.

To transfer Mod III cassette files to your PC, write a short M3-BASIC program to copy them to a TRSDOS 1.3 diskette, and use LINK3PC to transfer the disk files to an MS-DOS diskette.

ERROR 23 - DATA statement encountered at the end of a program line: If a DATA statement is not the first statement on a line and if the data items are alphanumeric and not delimited with quotes, then embedded spaces were removed. Review the M3-BASIC and GW-BASIC program lines and insert spaces as required.

ERROR 24 and 25 reserved.

ERROR 26 - CHR$() or STRING$(,) statement encountered: Both M3-BASIC and GW-BASIC use these statements in conjunction with the PRINT statement as an alternate method of displaying numbers, letters and symbols on the screen and to toggle certain video and system functions. For example, PRINT CHR$(65) would display an "A" on both the Mod III or PC screen. Thus, little change is necessary to CHR$(X) which display characters. However, to move the cursor to the upper left corner of the screen, a video function, M3-BASIC uses PRINT CHR$(28) while the GW-BASIC equivalent is LOCATE 1,1 (see Appendix F for more details).

ERROR 27 **OPEN statement encountered:** OPEN statements are flagged to aid you in checking filespec syntax and to alert you to differences in the default values for random files.

Filespecs: When OPENing files in either M3-BASIC or GW-BASIC you may specify a file extension and source drive. In M3-BASIC the extension is preceded by a slash, and the source drive is a number and appears to the right. For example, you might find a line like this:

```
5000        OPEN "I",1,"ROOTNAME/EXT:#"
```

where EXT is the extension and # is a number from 0 to 3.

In GW-BASIC the extension is preceded with a period, and the source drive is a letter which appears to the left. Under GW-BASIC, the above OPEN statement must be changed to:

```
5000        OPEN "I",1,"L:ROOTNAME.EXT"
```

where EXT is preceded by a period and "L" is the letter A, B, or C.

Both BASICs allow use of string variables to designate filespec. If you find a line like this:

```
4000        OPEN "I",1,A$
```

you must search elsewhere in your program and locate the place(s) where the filespec is assigned to A$. Look for a line containing a command like:

```
A$="ROOTNAME/EXT:#" or A$="ROOTNAME"+"/"+"EXT"+":#"
```

and change to: `A$="L:ROOTNAME.EXT" or A$="L:"+"ROOTNAME"+"."+"EXT"`

Random buffer size - default value: Unless specified, in M3-BASIC the buffer size of a random file defaults to 256 bytes. In GW-BASIC the default size is 128 bytes. The M3-BASIC OPEN statement:

```
5000        OPEN "R",1,"FILE/EXT"
```

needs to be changed to:

```
5000        OPEN "R",1,"FILE.EXT",256
```

For more complete details about the differences between M3-BASIC and GW-BASIC OPEN statements, please refer to your respective BASIC manuals. Also, be sure to review the LOF(buffer) statement.

ERROR 28 - RND( ) statement encountered: There are a few differences between the RND commands of M3-BASIC and GW-BASIC.

In M3-BASIC: RND(Ø) returns a decimal number between Ø and 1, and
RND(X) returns an integer between 1 and X inclusive, where X is a positive integer less than 32768.

In GW-BASIC: RND returns a decimal number between Ø and 1, and
RND(Ø) repeats the last random number presented.

The following lines show RND instructions as they might appear in M3-BASIC programs and their GW-BASIC equivalents:

```
          M3-BASIC                      GW-BASIC equivalent
3Ø    Z=RND(1Ø)                  3Ø    Z=INT(RND*1Ø)+1
4Ø    PRINT RND(5)-1             4Ø    PRINT INT(RND*5)
5Ø    K=RND(Ø)                   5Ø    K=RND
```

ERROR 29 - RANDOM statement encountered: Both M3-BASIC and GW-BASIC have statements to reseed the random number generator. M3-BASIC uses the RANDOM statement and requires no parameters. GW-BASIC uses RANDOMIZE, which may be followed by an integer value in the range of -32768 to 32767. If the optional value is omitted, when RANDOMIZE is executed the program will pause and request it. To avoid this nuisance but still guarantee a different sequence of pseudo random numbers each time the program is run, we suggest you MERGE the R651ØØ.ASC subroutine. Insert a GOSUB to this routine early in your program and replace each occurrence of: RANDOM
with: RANDOMIZE FNSEED

R651ØØ.ASC defines SEED as a function of TIME$ and returns a number in the required integer range (see Appendix G).

ERROR 3Ø - USING statement encountered: This statement is used in conjunction with PRINT and LPRINT to format displayed and printed data. The general syntax for the USING statement is as follows:

(L)PRINT USING **format string**;expression(s)

where format string may be a string literal or string variable.

Special characters used in the format string produce different results (see LPRINT USING and PRINT USING in your BASIC manuals for details). With two exceptions, the special characters used by both BASIC's are identical and no manual change will be required. The two exceptions are described below:

1. String field delineation: M3-BASIC uses two percent signs and GW-BASIC uses two backslashes.
2. Exponential notation: M3-BASIC uses four up-arrows (shown as [[[[) and GW-BASIC uses four carets (^^^^).

Note that CV3 automatically corrects these two exceptions if they occur in a USING statement where the format string is a string literal. For example, the M3-BASIC lines on the left will automatically be revised to the GW-BASIC lines on the right:

```
          M3-BASIC                          GW-BASIC
1Ø PRINT USING"%    %";A$            1Ø PRINT USING"\    \";A$
2Ø PRINT USING"##.##[[[[";X          2Ø PRINT USING"##.##^^^^";X
```

However, CV3 does not make changes when the format string is a string variable. If you find something like this:

1ØØ PRINT USING F$,A$ or 2ØØ PRINT USING E$,X

you must search elsewhere in your program, locate the place(s) where the format string is initialized and revise as above.

ERROR 31 - KILL, LOAD, or RUN statement encountered: These three statements are flagged because they require or optionally may be followed by a filespec in the form of a string literal or string variable.

If a string literal follows one of these statements, review it and make any necessary changes. If the filespec contains an extension, change the slash preceding it to a period. For example:

M3-BASIC	GW-BASIC
4ØØ RUN"NEWPROG/BAS"	4ØØ RUN"NEWPROG.BAS"

If the string literal designates a disk drive, remove the drive number and colon, and then insert the appropriate GW-BASIC drive letter and colon in front of the filespec. For example:

M3-BASIC	GW-BASIC
5ØØ KILL"OLDFILE:1"	5ØØ KILL"B:OLDFILE"

Warning: If a M3-BASIC filespec does not contain a drive number, M3-BASIC first looks for it on drive :Ø. If it is not located there, M3-BASIC continues to search for it on consecutively higher numbered drives. GW-BASIC interprets the lack of a drive specification to mean that the filespec is located on the current system default drive (usually drive A: on floppy disk systems and drive C: on hard disk systems). If the filespec is not on the system drive, program operation is aborted with a "file not found" error. You cannot assume that a M3-BASIC filespec without a designated drive number means that the program expected to find it on drive :Ø. Further, the drive letter you assign will depend on your PC's hardware configuration.

If KILL, LOAD or RUN is followed by a filespec in the form of a string variable, then you must search elsewhere in your program, locate the place(s) where the string variable is initialized and revise accordingly as per above.

ERROR 32 - LOF( ) statement encountered: There is a major difference between the LOF( ) commands in M3-BASIC and GW-BASIC.

The M3-BASIC LOF( ) statement returns the last record number in a random file. It was very useful when reading random files sequentially or when adding records to an existing file. For example:

```
3Ø REM Read file sequentially
4Ø FOR I=1 TO LOF(1)
5Ø    GET 1,I
6Ø    <<Process Record>>
7Ø NEXT
```

```
2ØØ REM Adding new records
21Ø X=LOF(1)+1
22Ø PUT 1,X
23Ø RETURN
```

The GW-BASIC LOF( ) statement returns the total number of bytes in a random file - rounded upward to a multiple of 128 (unless the file was created by the MS-DOS EDLIN utility). The GW-BASIC LOF( ) statement is only useful with a random file composed of 128 or 256 byte records.

If the M3-BASIC program being converted accesses random files with 128 or 256 byte records, just replace each occurrence of "LOF(buf#)" with "LOF(buf#)/rec.size". For example, if record size is 256 then line 4Ø above should be changed to:

```
4Ø FOR I=1 TO LOF(1)/256
```

and line 21Ø should be changed to:

```
21Ø X=LOF(1)/256+1
```

If the M3-BASIC program accesses random files that do not have 128 or 256 byte records it is still possible to make use of the LOF( ) statement by reformatting the data file to comply with the GW-BASIC's record size limitations. Reformatting such a file may be accomplished by the program below. It reads a record from the original file, expands it to 128 or 256 bytes and writes it to a new file. Note that the sample program illustrated here contains no error-trapping and should be used with extreme care (be sure to make a backup copy of all files prior to use).

```
1Ø 'EXPANDIT.ASC - Mod III BASIC file reformat program.
12 '               Run on Mod III before transferring file to PC
15 CLEAR 5ØØ:DEFINT A-Z:CLS
2Ø LINE INPUT"Original M3 filename, drive#, rec.size ";M3$,D1$,F1
25 LINE INPUT"Expanded M3 filename, drive#, rec.size (128 or 256) ";PC$,D2$,F2
3Ø OPEN "R",1,M3$+":"+D1$,F1:FIELD 1,(F1) AS A$
35 IF F2=256 THEN OPEN "R",2,PC$+":"+D2$ ELSE OPEN "R",2,PC$+":"+D2$,F2
4Ø FIELD 2,(F1) AS B$,(256+128*(F2=128)-F1) AS PD$:LSET PD$=" "
45 FOR I=1 TO LOF(1):GET 1,I:LSET B$=A$:PUT 2,I:NEXT:CLOSE:END
```

The advantage of restructuring the data file in this manner is that manual revisions to the converted program are minimal - just replace each occurrence of LOF( ) with LOF( )/128 or LOF( )/256. The disadvantage is that wasted space is built into each data record and there is the possibility of attempting to generate a file too large to fit on a Mod 1I1 diskette.

Other techniques may be used to avoid wasting record space such as (1) blocking or packing several small data records into one 128 or 256 byte record, and/or (2) creating a small one record mini file to keep track of the number of records on a data file. Refer to your BASIC manuals for additional information.

ERROR 33 - INPUT@ statement encountered: INPUT@ is a DOS PLUS enhancement to M3-BASIC. CV3 flags this statement but does not alter it. INPUT@ is roughly equivalent to a combination of PRINT@ and INPUT with parameters to delimit the number and type of characters entered by the user (refer to your DOS PLUS manual for details). Depending on the number of times the INPUT@ statement appears and the parameters used, you may want to substitute with LOCATE and INPUT, or replace each one with a GOSUB to a generalized subroutine that could be "tailored" for each specific INPUT@ by passing the screen address and parameters via subroutine variables. In either case, you may want to MERGE R65Ø3Ø.ASC with your program to convert the specified screen address to GW-BASIC LOCATE coordinates. Refer to ERROR 6 and Appendix G for additional information.

ERROR 34 - FIELD statement encountered: This is a warning error. CV3 attempts to insert a space before and after each occurence of "AS" in a FIELD statement. Review the flagged FIELD statement to be sure (1) that all required spaces were inserted, and (2) that CV3 did not incorrectly segment a variable being used to define a field's length or name.

Appendix A SUGGESTIONS FOR BETTER CONVERSIONS

While preparing the Mod III program for conversion there are a few steps you can take to avoid problems and further reduce your manual conversion effort.

1. Before saving your Mod III program in ASCII, remove all "invisible" line feeds that occur in the last position of physical program lines. They are easily found and eliminated by using EDIT and removing the last character of any physical program line that precedes a blank line in your program listing.

2. You may want to insert REM statements to identify the ASCII version:

```
10 REM*************************************************************
20 REM* Program Name:_____________________ ASCII Version   *
30 REM* TRSDOS 1.3 disk format.  Date saved in ASCII__/__/__  *
40 REM*************************************************************
```

3. To eliminate the possibility of any EOF handling problems when using the LINK3PC file transfer program, you may want to add a few dummy REM statements to the end of your Mod III program before saving it in ASCII. For example:

```
65500 REM *      Dummy REM - Delete After Conversion          *
65501 REM *      Dummy REM - Delete After Conversion          *
      "
```

4. After SAVEing the ASCII version, LOAD it under TRSDOS 1.3. If you get any "direct statement in file" errors, LOAD the original Mod III program, split the offending line, and SAVE the program in ASCII again.

5. GW-BASIC space delimiters, menu options J and K, and some GW-BASIC replacements will add several characters to the length of converted program lines. If space is not available, some portion of a line may he improperly converted or indented. This will require manual attention.

GW-BASIC reserves 5 spaces for line number and 1 space for the space following the line numher leaving a maximum of 249 characters for the body of the line. To locate long lines, do a visual scan or run the following Mod III program after saving the ASCII version:

```
10 CLEAR 1000:
15 CLS:LINE INPUT"Enter ASCII filespec - 'PROGNAME/ASC:#' ";F$
   OPEN "I",1,F$
20 IF EOF(1) THEN CLOSE:END
             ELSE LINE INPUT#1,A$:S$=LEFT$(A$,6)
30 S=INSTR(S$," ")+1:L=LEN(MID$(A$,S))
40 IF S=1 OR VAL(S$)=0 THEN PRINT S$" FORMAT ERROR":GOTO 20
50 IF L>249   THEN PRINT S$" > 249 **** TOO LONG ****":GOTO 20
60 IF L>200   THEN PRINT S$" > 200, May be too long":GOTO 20
70 IF L>150   THEN PRINT S$" > 150, May be too long for options J, K"
80 GOTO 20
```

6. The CV3 program ignores line 0 in M3-BASIC programs. If by chance any of your programs contain a line 0, simply type it back in after conversion or change it to another line number before saving it in ASCII.

Appendix B M3-BASIC PEEKs and POKEs

PEEKs and POKEs - GENERAL: The PEEK instruction returns the CHR$ code located in the address PEEKed and the POKE instruction stores a CHR$ code in the address POKEd. For example: X=PEEK(Y) sets X equal to the decimal equivalent of the binary value stored at Y, and POKE Y,X stores the binary equivalent of X in memory position Y.

The conversion program flags all PEEKs and POKEs but does not attempt to convert them since they require detailed manual inspection.

M3-BASIC used PEEK/POKE instructions for three reasons: (1) as an alternate method of displaying information on the screen and/or "reading" the screen, (2) to monitor or change certain system status codes, and (3) to store machine language subroutines in upper memory for later access with a USR statement.

PEEKs and POKEs to the Screen: Locations 15360 to 16383. Often you will be able to determine that the address used in a PEEK or POKE falls within this range, usually because either the address is a constant or the nearby program lines set a variable within that range.

```
460 FOR I=15360 TO 16383:POKE I,191:NEXT  or something like
470 FOR I=0 TO 1023:POKE 15360+I,191:NEXT
```

Lines 460 and 470 reveal the technique of POKEing to the Screen.

If you can make this determination:

For PEEKs: Remove the PEEK instruction and replace it with:

```
ZOG%=(address):GOSUB 65000
```

Subroutine 65000 returns the PEEKed value in variable ZIC%. The best way to illustrate the manual conversion is with a few examples. M3-BASIC lines are on the left and the required GW-BASIC substitutes are on the right.

M3-BASIC PEEKs	GW-BASIC replacements
X=PEEK(1564Ø)	ZOG%=1564Ø:GOSUB 65ØØØ:X=ZIC%
CLS:Y=PEEK(I+1Ø):PRINT	CLS:ZOG%=I+1Ø:GOSUB 65ØØØ:Y=ZIC%:PRINT
PRINT PEEK(1536Ø+64*L)	ZOG%=1536Ø+64*L:GOSUB 65ØØØ:PRINT ZIC%

For POKEs: Remove the POKE instruction and replace it with:

```
ZOG%=(address):ZUC%=(value):GOSUB 652Ø
```

M3-BASIC POKEs	GW-BASIC replacements
POKE 1536Ø,191	ZOG%=1536Ø:ZUC%=191:GOSUB 652Ø
CLS:POKE I,V:PRINT"--	CLS:ZOG%=I:ZUC%=V:GOSUB 652Ø:PRINT"--

The GOSUBed routines for PEEK and POKE to the screen are contained in R65000.ASC and listed in Appendix G.

PEEKs and POKEs to monitor/change system status: M3-BASIC uses some locations in the mid to upper 16ØØØ's to initialize, update, and monitor certain system conditions. M3-BASIC allows programs to PEEK these locations to check status and to POKE them to change status. One source of information about the RAM location of these codes and what they control is the green and white reference card which was once included with the purchase of a cassette based Mod III. The following describes some of most often used locations and some of their GW-BASIC solutions.

16Ø9 - Caps Lock Switch: Prior to accepting keyboard input, a M3-BASIC program could set this switch. POKEing a Ø allowed an upper and lower case keyboard input, and POKEing a non-zero value forced all upper case. Programs toggle this switch to save the user the trouble of pressing <SHIFT>Ø (the Mod III equivalent to the PC <CAPS> key) and to simplify program logic required to test user responses. You may see lines such as these:

```
25ØØ  POKE 16Ø9,1:INPUT"Type YES/NO and press <ENTER>";A$
252Ø  A$=LEFT$(A$,1):IF A$="Y" THEN ....
```

A better GW-BASIC response would be:

```
25ØØ  INPUT"Type YES/NO and press <ENTER>";A$
252Ø  A$=LEFT$(A$,1):IF A$="Y" OR A$="y" THEN ....
```

16412 - Cursor blink switch: If the M3-BASIC program POKEs a non-zero value here, the program is telling the system to stop blinking the cursor. At this moment we are not aware of a GW-BASIC equivalent.

	M3-BASIC	GW-BASIC
Turn on:	3ØØ POKE 16412,Ø	normal
Turn off:	4ØØ POKE 16412,1	n/a

16416, 16417 - Cursor address: If the M3-BASIC program contains a line like this:

```
3456 X=(PEEK(16417)-6Ø)*256+PEEK(16416)
```

it is storing the cursor's M3-BASIC PRINT@ address in variable X. Replace with:

```
3456 X=(CSRLIN-1)*64+POS(Ø)-1
```

so that your converted program will generate identical M3-BASIC PRINT@ addresses. Subroutine 65Ø3Ø will then convert PRINT@X statements to the proper GW-BASIC LOCATE coordinates.

16419 - Cursor character: If the M3-BASIC program contains a line like this:

```
75Ø POKE 16419,191
```

it is changing the cursor character to one of the graphics characters. GW-BASIC has a more limited ability to alter cursor character makeup. Although our GW-BASIC manuals claim that various widths of cursors can be specified by the "start" parameter of LOCATE, at the time of this writing we were unable to change the Tandy 2ØØØ cursor as described. Consult your GW-BASIC manual.

16424 - Maximum Lines/Page: If the M3-BASIC program contains this PEEK or POKE it is setting or checking the maximum number of lines per page. It is generally used at the beginning of a program which is printing reports on special forms with less than 66 lines per page. A simple GW-BASIC approach is to count the number of lines printed and LPRINT the necessary number of blank lines instead of using LPRINT CHR$(12) to skip to the top of the next form. Set a variable (MX) to the number of lines per page and add 1 to another variable (LC) with each LPRINT (remember to account for heading lines). Then substitute for each LPRINT CHR$(12) with a GOSUB to a routine such as:

```
654ØØ FOR I=1 TO MX-LC:LPRINT:NEXT:RETURN   'to move to top-of-form.
```

16425 - Lines printed plus one: If the M3-BASIC program is PEEKing this location, it probably is doing so before or after an LPRINT to see if a printed listing is nearing the bottom of the current page and needs to skipped up to the top of the next page. For example:

```
6ØØ IF PEEK(16425)>53 THEN LPRINT CHR$(12)
```

You can substitute by using the same approach previously mentioned for address 16424. Record the number of lines printed with a variable (LC) and after each LPRINT employ one of the following tests:

```
for standard paper (66 lines):      6ØØ IF LC>54 THEN LPRINT CHR$(12)
for non-standard paper:             6ØØ IF LC>## THEN GOSUB 654ØØ
```

16548, 16549 - Mod III program's starting position: These two locations contain the address of where a program begins LOADing into RAM. If a M3-BASIC program POKEs these locations, it is probably raising these pointers so that the next program(s) LOADed will have space reserved for a machine language subroutine. You may be able to delete this line and reserve space by requesting an extra file when entering BASIC. Then use the GW-BASIC VARPTR(buffer#) and DEF SEG commands to access it.

16561, 16562 - Memory size address holders: If the program is POKEing values into these locations, it is most likely lowering Mem Size to reserve space for a machine language subroutine. Substitute the GW-BASIC CLEAR, ##### command.

16913 - Cassette Baud Rate: If the program is POKEing this location it is changing the cassette baud rate. See ERROR 22 in Chapter 3 for assistance.

16916 - Screen Scroll Protect: If the M3-BASIC program is POKEing this location it is preventing the top few lines (Ø-7) from scrolling off the display. Although there is no equivalent GW-BASIC instruction, you can achieve a similar result by deleting the POKE and inserting a line like the following after the line which is causing the display to scroll:

```
####  IF CSRLIN = 17 THEN:LOCATE R,1:
      WHILE CSRLIN<17:PRINT STRING$(8Ø,32):WEND:LOCATE R,1
```

where R equals the number of the last line protected plus one.

16919 - Time/Date: If the M3-BASIC program is PEEKing or POKEing this location it is either accessing or setting the system time and/or date. Substitute with DATE$ and TIME$.

POKEs for storing machine language Subroutines: If the M3-BASIC program is POKEing several bytes of numeric data into contiguous memory locations, it is probably storing a machine language subroutine for later access with a USR statement. The machine language subroutine must be rewritten in PC assembler; the DATA statements used to store the routine will need to be revised with the decimal equivalents of the hex code generated by the recompilation; and the DEF USR address must be revised with the proper segment offset.

Appendix C CMD" " REPLACEMENTS

This table lists several M3-BASIC CMD" " statements and their GW-BASIC equivalents (if any). For additional information refer to the Mod III Disk System Owners Manual, the Mod III DISK BASIC reference card, and your GW-BASIC manual.

M3-BASIC CMD" "	Function	GW-BASIC equivalent
CMD"A"	Returns to DOS, displays "OPERATION ABORTED"	SYSTEM
CMD"B","ON/OFF"	Enable or Disable BREAK key	n/a
CMD"C"	Delete REM's and/or spaces	CV3 automatically deletes unnecessary spaces; use option L to remove REM's
CMD"D:X"	Displays directory (where X=Ø,1,2,3)	FILES"X:" (where X=A,B, or C)
CMD"E"	Displays last DOS error	PRINT ERR and refer to Appendix E
CMD"O",#,array$(begin)	Sort array$	n/a[1]
CMD"P",etc.	Returns printer status	See Appendix D
CMD"R"	Turns on clock display	n/a
CMD"S"	Returns to TRSDOS	SYSTEM
CMD"R"	Turns off clock display	n/a
CMD"X",etc.	Xref program line numbers, variables, etc.	n/a
CMD"Z","ON/OFF"	Turn on/off echo of display to printer.	n/a

[1]For sorting just a few items a simple BASIC bubble sort may suffice. For large arrays use a machine language subroutine.

```
64990 '** Sample GW-BASIC bubble sort **
64991 '* Sorts elements in array IZ$( ) - pass starting element
64992 '* in IB, no. of elements to sort in IE. Destroys IZ, I9
65000 I9=1:WHILE I9:I9=0:FOR IZ=IB TO (IB+IE-2):
      IF IZ$(IZ)>IZ$(IZ+1) THEN SWAP IZ$(IZ),IZ$(IZ+1):I9=1
65010 NEXT IZ:WEND:RETURN
```

Appendix D OTHER CONVERSION CONSIDERATIONS: Subtle Differences

INPUT statement: GW-BASIC nulls the value of string and numeric variables used in INPUT statements. In M3-BASIC, string or numeric variables used in INPUT statements retain their previous value unless some other value is entered by the user. In the following short program, if the user presses <ENTER> in response to line 2Ø, X will equal 35 with M3-BASIC but Ø with GW-BASIC.

```
1Ø X=35
2Ø INPUT"CHANGE";X
3Ø PRINT X
```

In some situations this difference between GW-BASIC and M3-BASIC can cause serious problems. For example, suppose you wrote a M3-BASIC program to create and update a random disk file containing the names and phone numbers for members of a local club. Suppose that the coding below is your "CHANGE ROUTINE" and you have just GOSUBed it.

```
2999   REM ** CHANGE ROUTINE **
3ØØØ   GOSUB 1ØØØØ   'GO ACCEPT DISK RECORD NUMBER
3Ø1Ø   GOSUB 11ØØØ   'GO GET DISK RECORD
3Ø2Ø   GOSUB 12ØØØ   'GO UNLOAD BUFFER INTO BN$, BP$
3Ø3Ø   PRINT"CHANGE ROUTINE
       To change a field, type the change and press <ENTER>.
       To leave unchanged, just press <ENTER>."
3Ø4Ø   PRINT"Name is "BN$"  Type change ";:INPUT BN$
3Ø5Ø   PRINT"Phone # is "BP$" Type change ";:INPUT BP$
3Ø6Ø   GOSUB 13ØØØ    'GO RELOAD BUFFER
3Ø7Ø   GOSUB 14ØØØ    'GO PUT UPDATED RECORD
31ØØ   RETURN
```

This routine will not produce the desired results in GW-BASIC, because pressing <ENTER> at line 3Ø4Ø or 3Ø5Ø will destroy the contents of the respective field(s) on the disk record. INPUT statements should be examined to determine whether successful Mod III results depended on retained value(s).

KEYBOARD TYPE-AHEAD: The PC has keyboard type-ahead, a feature which allows you to load the keyboard input buffer even when your program isn't asking for a response. Although this can be a helpful feature at times, it can be an inconvenience at others. For example, suppose your program is displaying user instructions and pauses between frames with 'INPUT"Press <ENTER> to continue";A$.' If the user pressed <ENTER> several times in succession before this prompt, or experienced keyboard bounce after his last <ENTER>, the screen will not pause between frames and as a result some of the instructions will be missed.

Inserting a simple WHILE/WEND loop before the INPUT pause will clear the buffer and eliminate the problem. For example:

```
3ØØØ WHILE INKEY$<>"":WEND:INPUT"Press <ENTER> to continue";A$
```

TIME DELAYS: FOR/NEXT loops are often used to pause program execution. Because PCs execute much faster than the Mod III, you may find it necessary to increase the FOR/NEXT limit. For example:

```
1Ø FOR I=1 TO  5ØØ:NEXT:RETURN 'Approx. one second delay on Mod III
1Ø FOR I=1 TO 2ØØØ:NEXT:RETURN 'Approx. one second delay on PC
```

PRINTER STATUS & TIME-OUT: M3-BASIC "hangs up" when trying to execute an LPRINT if the printer is not ready. The only way to continue processing is to activate the printer or press <BREAK>. On the PC's we tested, we found the following when trying to execute an LPRINT:

1. Printer not attached: GW-BASIC responded with
   "Device Timeout" (ERR = 24) after about 6 seconds.
2. Printer attached, but not on: program execution continues.
3. Printer attached, on, but not selected: Program hangs up.

Some experimentation is suggested for your particular PC and printer combination. We found that an ON ERROR GOTO routine would only trap case number one above. You may want to insert a prompt prior the the print routine like this:

```
4ØØØ   WHILE INKEY$<>"":WEND:INPUT"Ready Printer and press <ENTER>";A$
4Ø2Ø   REM -- Report routine goes here
```

MEMORY SIZE: A 48K Mod III has about 38K of RAM available for program storage, while a PC has only 25K available. Hence, a large M3-BASIC program that is converted may not fit into GW-BASIC's storage area or may possibly run out of memory on execution. To estimate the amount of memory the converted program will require: LOAD and RUN the M3-BASIC program just long enough to execute DIM's, load arrays, and initialize variables. Then press <BREAK>, type ?MEM and press <ENTER>. Subtract the number returned from 3828Ø and add a rough estimate for GW-BASIC space delimiters, etc. If the answer is close to 25ØØØ, the program is probably too large, and you should review the GW-BASIC CHAIN and MERGE commands and implement program segmentation. Note: Do not select menu options J and K (insertion of down-arrows) when converting programs that will require segmentation.

COMMAS IN PRINT STATEMENTS: With M3-BASIC, commas used in PRINT statements cause cursor "jumps" to positions 16, 32, and 48. In GW-BASIC, commas cause "jumps" to positions 14, 28, 42, 56, and 7Ø. Thus, a statement such as:

```
                    PRINT W,X,Y,Z
must be changed to  PRINT W;TAB(16)X;TAB(32)Y;TAB(48)Z
```

FOR/NEXT LOOPS: In general, M3-BASIC and GW-BASIC execute FOR/NEXT loops in the same manner. The two differ, however, in the fact that GW-BASIC does not allow conditional NEXT statements (see ERROR 21 description) and M3-BASIC always executes a FOR/NEXT loop at least once while GW-BASIC will bypass the loop if the product of the initial value of the loop control variable and the step increment is greater than the product of the final value of the loop control variable and the step increment. For example:

```
5ØØ   FOR I=1 TO 5 STEP -1
51Ø     PRINT"-1 is greater than -5, loop is bypassed by GW-BASIC"
52Ø   NEXT
```

Note that the need for program modification is not always implicit in the FOR statement. If one or more of the loop control values is a variable, as in the following:

```
4ØØ FOR I=1 TO 5 STEP Z
```

you will need to review program logic and substitute values assigned during execution to make this determination.

Appendix E M3-BASIC/GW-BASIC ERROR CODE XREF

This table describes error codes returned by M3-BASIC and GW-BASIC. While both BASIC's assign error codes to ERR, M3-BASIC assigns a false code which must be interpreted to determine the true error code and GW-BASIC assigns the true code itself.

M3-BASIC "false code" ERR	M3-BASIC "true code" ERR/2+1		GW-BASIC "true code" ERR	Description
Ø	1	=	1	NEXT without FOR
2	2	=	2	Syntax error
4	3	=	3	RETURN without GOSUB
6	4	=	4	Out of DATA
8	5	=	5	Illegal function call
1Ø	6	=	6	Overflow
12	7	=	7	Out of memory
14	8	=	8	Undefined line
16	9	=	9	Subscript out of range
18	1Ø	=	1Ø	Redimensioned array
2Ø	11	=	11	Division by zero
22	12	=	12	Illegal direct
24	13	=	13	Type mismatch
26	14	=	14	Out of string space
28	15	=	15	String too long
3Ø	16	=	16	String formula too complex
32	17	=	17	Can't continue
34	18	+1	19	No RESUME
36	19	+1	2Ø	RESUME without error
38	2Ø	+1	21	Unprintable error
4Ø	21	+1	22	Missing operand
42	22	+1	23	M3=Bad file data PC=Line buffer overflow
Disk Errors:				
1ØØ	51	-1	5Ø	Field overflow
1Ø2	52	-1	51	Internal error
1Ø4	53	-1	52	Bad file number
1Ø6	54	-1	53	File not found
1Ø8	55	-1	54	Bad file mode
114	58	-1	57	Device I/O error
122	62	-1	61	Disk full
124	63	-1	62	Input past end
126	64	-1	63	Bad record number
128	65	-1	64	Bad file name
132	67	-1	66	Direct state. in file
134	68	-1	67	Too many files

Appendix F M3-BASIC CHR$() and STRING$(,) CODES

M3-BASIC uses ASCII codes from Ø to 255 to display characters and special symbols on the screen, to invoke video control functions and as "switches" to control the function of some ranges of codes that have more than one use.

ASCII CODES (32-127): These codes represent letters, numbers and commonly used text symbols. For example, PRINT CHR$(65) displays an "A" and PRINT CHR$(43) displays a "+" symbol. The majority of these codes produce the same result in M3-BASIC and GW-BASIC. If a program line was flagged by ERROR 26 and it contains only codes in this range you can ignore the flag.

ASCII CODEs (128-191): These codes display M3-BASIC block graphic characters. They can be reviewed in the Mod III Operation and BASIC Reference Manual or you can use this short routine to display them on the Mod III screen:

```
FOR I=128 TO 191:PRINT USING"### ";I;:PRINT CHR$(I):NEXT
```

The GW-BASIC block graphic set is limited, but the codes 177, 178, 219, 22Ø, 221, 222, 223, or 254 may provide worthy alternatives. For example, the M3-BASIC statements:

```
PRINT CHR$(191) and PRINT STRING$(64,191)
```

may be replaced with GW-BASIC's:

```
PRINT CHR$(219) and PRINT STRING$(64,219)
```

ASCII CODEs (192-255): Each of these codes has two exclusive functions controlled a by "switch." The switch, normally off, may be turned on by executing a PRINT CHR$(21) statement. Once set, it stays on until another PRINT CHR$(21) is executed.

When the switch is off, M3-BASIC treats the codes as space compression characters and skips the cursor "code less 192" spaces to the right. For example, PRINT CHR$(195) skips the cursor 3 positions.

```
PRINT"SAMPLE";:PRINT CHR$(195);:PRINT"SKIP"
```

or simply: `PRINT"SAMPLE"CHR$(195)"SKIP"`

displays: "SAMPLE...SKIP" (ignore quotes & periods).

For "small" skips, replace PRINT CHR$(X) by inserting a blank string with the required number of spaces:

```
PRINT"SAMPLE""  ""SKIP", or PRINT"SAMPLE  SKIP"
```

For "large" skips, replace PRINT CHR$(X) with:

```
PRINT SPACE$(X-192), or PRINT STRING$(X-192,32).
```

When the switch is on, M3-BASIC uses these codes to display special characters. These can be reviewed in the Mod III Operation and BASIC Reference Manual or you can display them on the screen with:

```
PRINT CHR$(21):FOR I=192 TO 255:PRINT USING"### ";I;:PRINT CHR$(I):NEXT
```

Note that we activated the special character mode by turning the switch on with PRINT CHR$(21).

Problems: If you know whether the PRINT CHR$(21) switch is on or off, substituting a GW-BASIC equivalent is rather easy. The only problem you may encounter is determining the status of the switch. This will require review of your program logic. Finally, remember to remove each PRINT CHR$(21) statement from your program after making the changes.

ASCII CODEs (Ø-31): Each code in this range serves two functions. When used in POKE statements they display special characters (refer to your M3-BASIC manuals). The primary use of these codes, though, is to invoke certain video control functions such as those described in the chart below.

M3-BASIC	Function	GW-BASIC equivalent.
PRINT CHR$(8)	Backspace cursor	LOCATE CSRLIN,POS(Ø)-1
PRINT CHR$(1Ø)	Linefeed & Car.Rtn.	same
PRINT CHR$(13)	Linefeed & Car.Rtn.	same
PRINT CHR$(14)	Turn on cursor	LOCATE ,,1
PRINT CHR$(15)	Turn off cursor	LOCATE ,,Ø
PRINT CHR$(21)	Switches special/ compression char.	remove after sub'ing for PRINT CHR$(192-255) Stmts.
PRINT CHR$(23)	Shifts to 32 char. mode (large size)	WIDTH 4Ø
PRINT CHR$(26)	Downward line feed	LOCATE CSRLIN+1,POS(Ø)
PRINT CHR$(27)	Upward line feed	LOCATE CSRLIN-1,POS(Ø)
PRINT CHR$(28)	Homes cursor	LOCATE 1,1
PRINT CHR$(29)	Crsr. to beg'g of line	LOCATE CSRLIN,1
PRINT CHR$(3Ø)	Clears to end of line	PRINT STRING$(81-POS(Ø),32)
PRINT CHR$(31)	Clears to end of scrn	see below.*

*Replace with: WHILE CSRLIN<17:PRINT STRING$(81-POS(Ø),32):WEND
You may want to make this a subroutine such as:
65ØØ WHILE CSRLIN<17:PRINT STRING$(81-POS(Ø),32):WEND:RETURN
and replace each occurrance of PRINT CHR$(31) with, in this case, GOSUB 65ØØ.

Appendix G MISCELLANEOUS GW-BASIC SUBROUTINES

The routines listed here are referred by other sections of this user manual and may be required in your converted program(s). They are provided in ASCII format on the diskette containing the conversion program and may be MERGEd to a converted program as required or entered via the keyboard. Since lines containing REM statements are not referenced, they may be deleted to conserve RAM.

These "subprograms" use line numbers above 64994 and uncommon variable names in an attempt to eliminate conflicts with your program. Nevertheless, please review your program listings before using these routines. If line numbers or variables require alteration, then the statements inserted by the conversion program, which GOSUB these routines and use their variables, will also need revision.

```
64995 '*************************************************
      '* R65000.ASC  Routines GOSUBed to replace screen *
      '* peeks 'n pokes.                                *
64996 '*   GOSUB 65000 for PEEKs. GOSUB 65020 for POKEs *
      '*************************************************
65000 ZOG%=ZOG%-15360:ZAG%=INT(ZOG%/64):ZOG%=ZOG%-64*ZAG%+1:
      ZAG%=ZAG%+1:ZIC%=SCREEN (ZAG%,ZOC%):RETURN
65020 ZAP%=CSRLIN:ZUP%=POS(0):GOSUB 65000:LOCATE ZAG%,ZOG%:
      PRINT CHR$(ZUC%);:LOCATE ZAP%,ZUP%:RETURN
65027 '*************************************************
      '* R65030.ASC - Calculates PRINT@s with variables.*
      '*************************************************
65030 ZAG%=INT(ZOG%/64):ZOG%=ZOG%-64*ZAG%+1:ZAG%=ZAG%+1:
      LOCATE ZAG%,ZOG%:RETURN
65095 '*****************************************************
      '* R65100.ASC - Randomize Seed: GOSUB near beginning
      '* of prog. or change line# and delete RETURN
65096 '*****************************************************
65100 DEF FNSD(X)=VAL(MID$(TIME$,X,2))*60^(5\(X+1)):
      DEF FNSEED=(FNSD(1)+FNSD(4)+FNSD(7))*.7585157-32768!:RETURN
65145 '*****************************************************
      '* R65150.ASC  Routine to display graphics characters *
      '* available from CHR$(169) thru CHR$(254)            *
65146 '*****************************************************
65150 CLS:LOCATE 4,10,0:FOR I=169 TO 254:
      IF (I>170 AND I<174) OR (I>223 AND I<240) OR (I>240 AND I<254)
      THEN 65152 ELSE PRINT USING"  ### ";I;:PRINT CHR$(I);:J=J+1
65152 IF J=5 THEN LOCATE CSRLIN+2,10:J=0
65154 NEXT:LOCATE ,,1:END
65165 '*************************************************
      '* R65170.ASC  Routine to draw double thinline *
      '* border around PC screen.  Revise to suit.   *
65166 '*************************************************
65170 RW%=CSRLIN:CL%=POS(0)'Save cursor position
65175 LOCATE 1,1:PRINT CHR$(201)STRING$(62,205)CHR$(187);
65176  FOR I=2 TO 15
65177   LOCATE I, 1:PRINT CHR$(186);:LOCATE I,64:PRINT CHR$(186);
65178  NEXT
65179 LOCATE 16,1:PRINT CHR$(200)STRING$(62,205)CHR$(188);
65180 LOCATE RW%,CL% 'Reposition cursor to original
65181 RETURN
```

Appendix H LINK3PC Program/File Transfer

This package includes a diskette labelled HYPERCROSS which contains programs to transfer TRS-80 Mod I/III programs and files to a PC. HXIII/CMD is for Mod III owners, and HXI/CMD is for Mod I owners. **The HYPERCROSS diskette is distributed in TRSDOS 2.3 Mod I single density format.** The procedure for making working copies and using the programs varies slightly depending on your hardware and DOS configuration.

Making a working copy: Boot your Mod I/III with a clean copy of your DOS in drive Ø. Supported DOS's include: TRSDOS 1.3, DosPlus 3.4 & 3.5, LDOS, MultiDos, NewDos 8Ø V2, TRSDOS 2.3, and TRSDOS 2.7.

1. Make a backup of your DOS system diskette.
2. Remove the original copy of DOS from drive Ø and store away.
3. Place the new backup copy of DOS in drive Ø.
4. Place the HYPERCROSS diskette in drive 1.
5. Transfer HXIII/CMD and/or HXI/CMD to your DOS diskette in drive Ø.
   Refer to your DOS manual for the required command. For example, with a Mod III and TRSDOS 1.3 use **CONVERT :1 :Ø**, and
   with a Mod I and TRSDOS 2.3 use **COPY HXI/CMD:1 :Ø**.
6. After successfully transferring HXIII/CMD or HXI/CMD to your DOS diskette, remove the HYPERCROSS disk from drive 1 and store away.
7. Backup the diskette in drive Ø for additional working copies.

Running HXIII/CMD: Boot your Model I/III with a DOS diskette containing HXI/CMD or HXIII/CMD in drive Ø.

1. Place the diskette containing the data file or ASCII program to be transferred to the PC in drive 1 (see page 2.3).
2. Copy the file or program to drive Ø.
   Type: **COPY NAME:1 :Ø** and press <ENTER>
   where **NAME** is the filename of the data file or ASCII program.
3. When COPY is finished, remove the diskette in drive 1 and store away.
4. Execute the HXIII or HXI program.
   Mod III owners, type: **HXIII** and press <ENTER>.
   Mod I owners, type: **HXI** and press <ENTER>.
5. When the "...Specify DOS type..." prompt and DOS menu appears, type the letter corresponding to your particular DOS.
6. When the function menu appears, type 4 and press <ENTER> to FORMAT an MS-DOS diskette in drive 1.
7. Place a blank diskette in drive 1 and press <ENTER>.
8. When the ">>" prompt appears, type: 2 and press <ENTER>. At the "Filename:" prompt, type: **NAME** and press <ENTER> where **NAME** is the filename used in step 2. The transferred file is automatically assigned the same NAME on the MS-DOS diskette. If the Mod I/III NAME included an /EXTension, the "/" is replaced with a ".".
9. At the "ASCII or Image file transfer (A, I or R)?" prompt:
   Type: A when transferring BASIC programs and ASCII data files.
   Type: I when transferring random data files with 256 byte records.
   Type: R### when transferring random data files containing records of less than 256 bytes. Substitute the actual record length for ### (e.g. to transfer a file of 64 byte records, type: RØ64).

When the ">>" reappears, press <ENTER> to return to the function menu or type 5 and press <ENTER> to exit the program. Remove the diskettes and refer to page 2.4.

Appendix I **IN CASE OF TROUBLE**

I. Using HYPERCROSS.

PROBLEM 1: HYPERCROSS diskette appears defective. Errors result from trying to load a program, and displaying a directory of the diskette is impossible.

Solution: The HYPERCROSS programs are supplied on a TRS-80 Mod I TRSDOS 2.3 data diskette. If you are not using a Mod I and TRSDOS 2.3, consult your DOS manual for the procedure for reading a TRSDOS 2.3 data diskette.

PROBLEM 2: One of the data files I want to transfer with HYPERCROSS is too large to fit on a system disk in drive :0. How can I get it on a PC diskette?

Solution: You will need to segment the file into two or more manageable pieces and use HYPERCROSS to transfer each piece. After transfer to the PC merge the segments.

II. Using CNV3TOPC.BAS.

PROBLEM 1: CNV3TOPC aborts or prints several "line too long" errors, or both. Also, when trying to list the ASCII version of the Mod I/III program with the MS-DOS TYPE command, "garbage" is displayed. A few reserved words (i.e. PRINT, GOTO, etc.) are legible but they are incorrect.

Solution: The Mod I/III program was not saved properly in ASCII format. Time and frustration can be avoided by making sure that a valid ASCII version has been saved before attempting to transfer and convert. The "A" option is required to SAVE an ASCII version of a program. The proper syntax is as follows: **SAVE"Program Name/Ext:0",A.** Note that the SECOND PAIR OF QUOTES IS MANDITORY and precedes the comma. Please follow the procedures described in the user guide on page 2.3 and Appendix A.

III. **Using a converted PC program.**

PROBLEM 1: After running a conversion with option "I" to completion, I can't LOAD the converted PC version for manual cleanup. BASIC displays a "file not found" error message.

Solution: Most likely when running CNV3TOPC you did not include an extension as part of the name you assigned to the converted PC program. GW-BASIC assumes the program named in the LOAD command has an extension of ".BAS" unless another extension is specified. To load a GW-BASIC program lacking an extension include a period after the name (e.g. to load a program called NAME, type LOAD"NAME."). Alternatively, you can use the MS-DOS REN (Rename) command, assign a ".BAS" extension and then LOAD the program.

PROBLEM 2: The converted program aborts with an "ILLEGAL function call" in a line containing an OPEN statement.

Solution: This problem is probably caused by the fact that some PC BASICs allocate only 128 bytes for random file buffers unless the "/S" option in used when BASIC is loaded. To allocate additional buffer space type: BASIC /S:256. BASICA users type: BASICA /S:256.

PROBLEM 3: The converted program aborts with "FIELD overflow in ..."

Solution: A FIELD statement is attempting to define a record larger than the default or specified value in its related OPEN statement. Be sure that all OPEN statements flagged during the conversion were revised according to instructions on page 3.6 (see Random buffer size - default value).

PROBLEM 4: Reports generated by converted programs print only 80 columns of information on a line and print the remainder on the next line.

Solution: Reports produced by GW-BASIC programs default to 80 columns unless a greater width is specified with the WIDTH command. Insert the following command somewhere near the beginning of the program:

```
WIDTH "LPT1:",255.
```

PROBLEM 5: Execution of a converted program aborts with a "Syntax error" message.

Solution: It is possible that a line was not edited properly during manual cleanup. Check the line closely to be sure you fixed errors flagged during the conversion.

If the line appears correct, the coincidence of a Mod I/III variable name being a GW-BASIC reserved word may have caused the trouble. Though Mod I/III BASIC recognized the first two alphanumeric characters in variable names, more could be used for readability. For example, the variable KEY, which was sometimes used as a pointer to random file records, is a GW-BASIC reserved word that may only be used as specified in the GW-BASIC manual. Check variable names in lines that otherwise appear to be correct. If by chance a variable begins with a GW-BASIC keyword, you will need to replace each occurrence of that variable in your program.